Heatmap Lab

By Upama KC

A heatmap is the visualization of the data table in terms of colors. Heatmap is a graphical representation of data in which individual values contained in a matrix and are represented by colors. It helps to recognize pattern within a data set visually by condensing multiple responses and predictor variables into one figure by highlighting similarities and/or differences between response and predictor variables. It has been used mostly in natural and biological sciences.

About data

Data contains eight observations (rows) in two different ecoregions on the microbial community and environmental variables (9 variables). First seven columns have information about soil physical properties and climate data of the sampling sites and remaining columns have a relative abundance of different microbial groups. However, variables could be anything from ecological, biological data set.

- Let's import dataset and look at it.

```
hm<- read.csv("heatmap.csv")
fix(hm)
head(hm)
str(hm)            #take quick look at the structure of the data, using the "str" command
row.names(hm)=hm$Sample            #creates row names
```

If you want to know more about heatmap in R, you can explore with

```
?heatmap
```

Create a heatmap

We need to install package "gplots" where we use function heatmpa.2

```
library(gplots)
library(RColorBrewer)
```

We need a numeric matrix for heatmap, so let's subset our data table "hm". After creating a matrix of sub-set of our data we need to scale and transpose the matrix. However, transposing depends on your choice. Scale() calculates the mean and standard deviation for the vector then scales each element by those values. It scales by subtracting the mean and dividing by the standard deviation. Scale (x, scale = FALSE) only subtract the mean (does not divide by the standard deviation). Scale() is especially important when we have multiple variables with different scales. For example, like with this data "hm" where variables like "BD" and "N" have a magnitude of 1 and variables like "Total.Bacteria" has an order of magnitude 100,000. Scale can be applied to rows or columns as per our need. We transpose matrix so that samples align to x-axis and different measurements to the y-axis in heatmap.

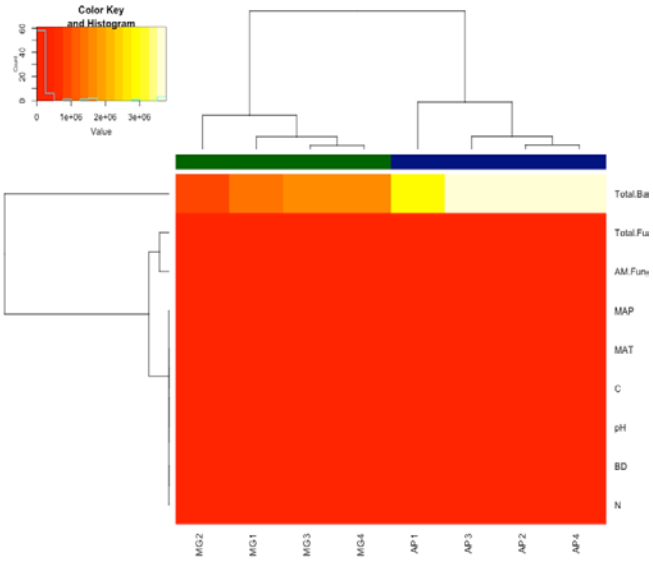
```
hm1= hm[,3:11]          # subset the data
hm_scaled=as.matrix(hm1)       # scale the matrix ( scale the data to mean=0 and sd=1)
hm_scaledt=t(hm_scaled)        # transpose the matrix
```

Create a sidebar for our heatmap that indicates ecoregion (treatment). For this, we must define the color that we want to use to differentiate ecoregion Aspen Parkland and mixed grassland will be navy and darkgreen respectively.

```
sidecols=c("navy", "navy", "navy","navy", "darkgreen", "darkgreen", "darkgreen","darkgreen")
heatmap.2(hm_scaledt, ColSideColors=sidecols, scale = "none",trace="none") #no scale
```

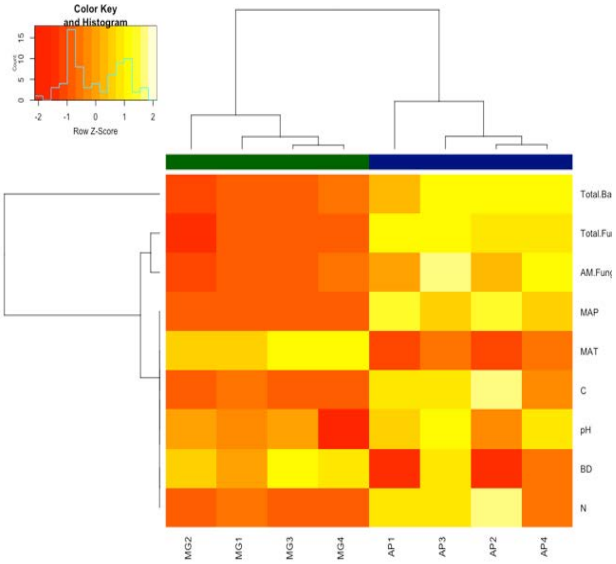
Here, red color represents the lowest value and yellow represent the highest value. The heatmap without scale is not informative. Indeed, “Total Bacteria” has very high values than any other variables which make all other variables look same with one color, red. Thus, once we normalize matrix using scale argument.

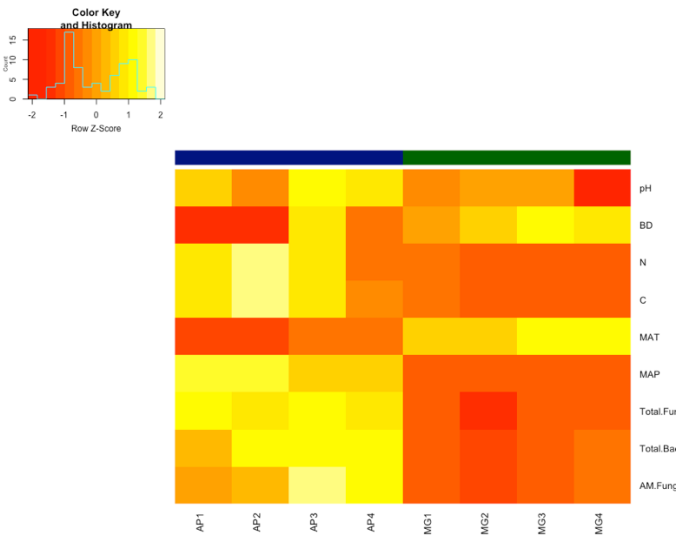
```
heatmap.2(hm_scaledt,
ColSideColors=sidecols, scale = "row",
trace="none" )  #scale by row
heatmap.2(hm_scaledt,
ColSideColors=sidecols, scale = "column",
trace="none" )   #scale by column
```



You might have noticed that the order of both rows and columns is different compared with our original “hm1” matrix. This is because of the corresponding dendrogram beside the heatmap. Heatmap performs reordering using clusterization: it calculates the distance between each pair of rows and columns and tries to order them by similarity. The default is to use dist() to calculate the distance matrix and hclust() for the hierarchical clustering that produces the dendrograms. You can avoid it and dendrogram and visualized raw matrix as well by using “Rawv” and “Colv” arguments as follow:

```
heatmap.2(hm_scaledt,ColSideColors=sidecols, Colv = NA, Rowv = NA, scale="row",
trace="none")
```





Custom layout of heat map

We can custom color, title and axis title with the usual main and xlab/ylab argument. We can also change labels with labRow/colRow and their size with cexRow/cexCol.

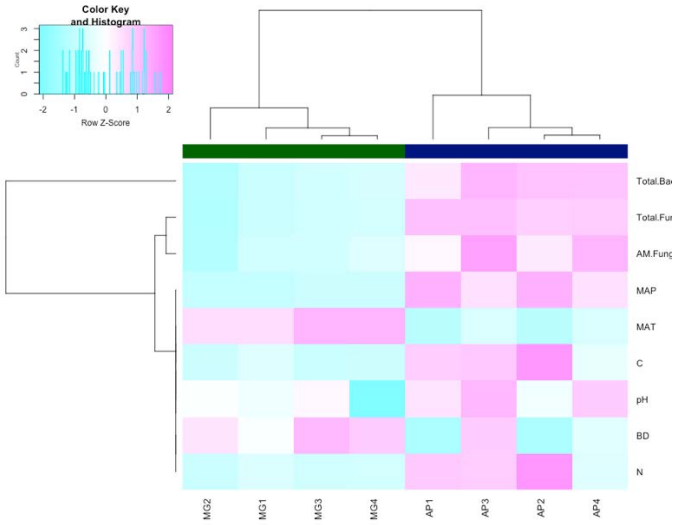
Color

1. Change the color of heat map by using native palette from R

   heatmap.2(hm_scaledt, ColSideColors=sidecols, scale = "row", trace="none",col=cm.colors(256))

2. Change the color by using Rcolorbrewer palette. RColor Brewer allows us to generate a custom colour palette for our heatmaps.

   library(RColorBrewer)
   colr = colorRampPalette(brewer.pal(8, "PiYG"))(25)
   heatmap.2(hm_scaledt,ColSideColors=sidecols, scale = "row", trace="none", col=colr)



Add title and axis label

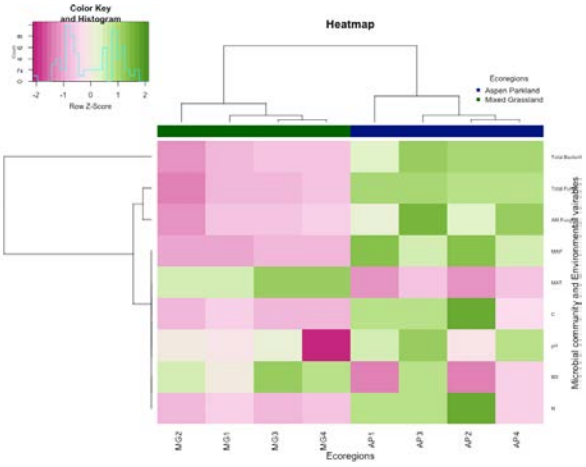
heatmap.2(hm_scaledt, ColSideColors=sidecols, scale = "row", trace="none", col=colr,cexRow=0.8, xlab="Ecoregions", ylab="Microbial community and Environmental vairables", main="Heatmap")

Add legend

We can add a legend that explains the sidebar colors manually. We will use coords as the legend coordinates, and define legend as below.

coords=locator(1)
legend(coords, pch=c(15,15), cex=0.8, bty="n", legend=c("Aspen Parkland", "Mixed Grassland"), title=c("Ecoregion"), col=c("navy", "darkgreen"))

Click on the heatmap figure where you want to place the legend.



Pretty heatmap with pheatmap

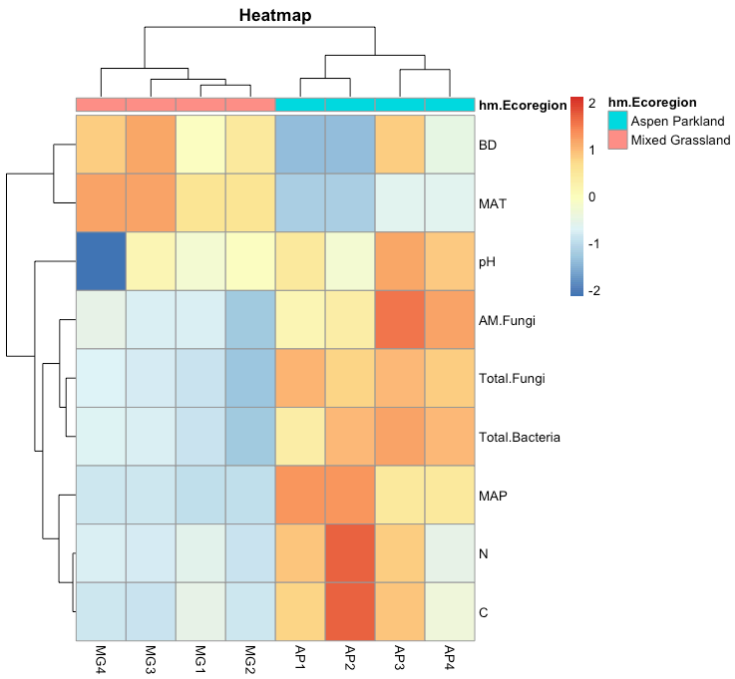
We can use the package “pheatmap” to create heatmap. This package simplifies script and comes with many functions which make it easy to create and manage heat plot. Pheatmap creates a similar heatmap as heatmpa.2 but arranges the samples differently. Pheatmap automatically creates an ecoregion legend while heatmap.2 could not. We are using the same data and matrix. First, annotating the row and/or columns to create sidebars that indicate ecoregion.

```
library(pheatmap)

gr <- data.frame(hm$Ecoregion) #creates ecoregion list
rownames(gr) = colnames(hm_scaledt) #adds row names from your data
fix(gr)

pheatmap(hm_scaledt, annotation = gr, cexRow = 0.5, scale="none", trace="none", main =
"Heatmap")

pheatmap(hm_scaledt, annotation = gr, cexRow = 0.5, scale="row", trace="none", main =
"Heatmap")
```



Try scaling by row and column and see how it changes the heatmap. Try ?pheatmap to see how you can customize your figure.

Correlation matrix heatmap: ggplot2

We will use the same data set and subset as a matrix but with a different name “dat”.

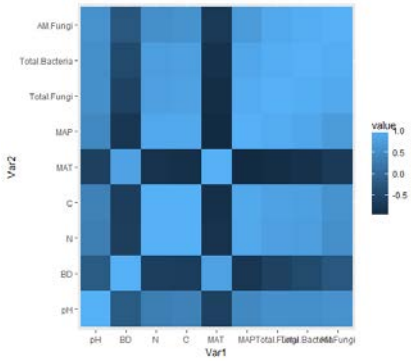
```
str(hm)
dat<- hm[, c(3:11)]
str(dat)
head(dat)
```

First, we need to compute the correlation matrix, which can be created using the R function cor()

```
cormap <- round(cor(dat), 2)
head(cormap)
```

Then, we need to generate the correlation heatmap with ggplot2 for that; package reshape is required to melt the correlation matrix

```
library(reshape2)
melted_cormap <- melt(cormap)
head(melted_cormap)
```



The function geom_tile()[ggplot2 package] is used to visualize the correlation matrix .

```
library(ggplot2)
ggplot(data = melted_cormap, aes(x=Var1, y=Var2, fill=value)) +
 geom_tile()
```

The matrix is not very nice because it has redundant information. We can visualize either upper or lower matrix and use the functions to set half of it to NA.

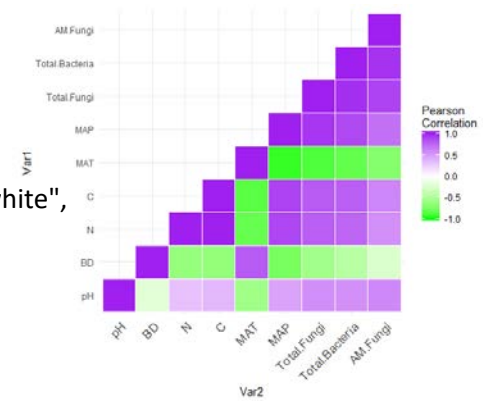
Get the lower/upper triangle of the correlation matrix

```
lower_tri<-function(cormap){cormap[upper.tri(cormap)] <- NA
 return(cormap)}
upper_tri <- function(cormap){cormap[lower.tri(cormap)]<- NA
 return(cormap)}
up_tri <- upper_tri(cormap)
up_tri
```

Now we will melt correlation data and drop the row with NA values. The function **coord_fixed()** ensures that one unit on the x-axis is the same length as one unit on the y-axis. The function **scale_fill_ gradients2** is used with the argument limit=c(-1,1) as correlation coefficients range from -1 to 1.

```
library(reshape2)
melted_cormap <- melt(up_tri, na.rm = TRUE)

library(ggplot2)
ggplot(data = melted_cormap, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "green", high = "purple", mid = "white",
 midpoint = 0, limit = c(-1,1), space = "Lab",
 name="Pearson\nCorrelation") +
 theme_minimal()+ theme(axis.text.x = element_text
```



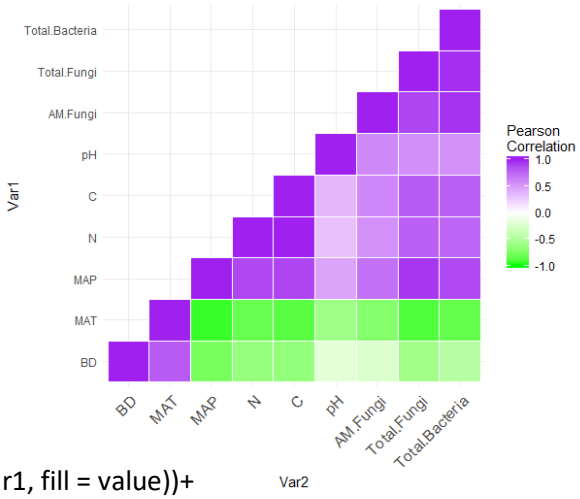
```
(angle = 45, vjust = 1,  size = 12, hjust = 1))+
coord_fixed()
```

Reorder the correlation matrix & customize final graph

We can reorder the correlation matrix according to the correlation coefficient. This is useful to identify the hidden pattern in the matrix. hclust for hierarchical clustering order is used in the example below where the correlation between variables is used as distance.

Helper function to reorder the correlation matrix:

```
reorder_cormap <- function(cormap) {
 dd <- as.dist((1-cormap)/2)
 hc <- hclust(dd)
 cormap <-cormap[hc$order, hc$order]}
```



Reorder and melt the correlation matrix

```
 cormap <- reorder_cormap(cormap)
up_tri <- upper_tri(cormap)
melted_cormap <- melt(up_tri, na.rm = TRUE)
```

Create the new heatmap:

```
ggheatmap <- ggplot(melted_cormap, aes(Var2, Var1, fill = value))+
geom_tile(color = "white")+
scale_fill_gradient2(low = "green", high = "purple", mid = "white",
midpoint = 0, limit = c(-1,1), space = "Lab",
name="Pearson\nCorrelation") +
theme_minimal()+
theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1))+
coord_fixed()
print(ggheatmap)
```

Customize and add correlation coefficients: Use **geom_text()** to add the correlation coefficients on the graph, use a **blank theme** (remove axis labels, panel grids and background, and axis ticks), and use **guides()** to change the position of the legend title:

```
 ggheatmap + geom_text(aes(Var2, Var1, label = value),
color = "black", size = 4) +
theme(
axis.title.x = element_blank(),
axis.title.y = element_blank(),
panel.grid.major = element_blank(),
panel.border = element_blank(),
panel.background = element_blank(),
axis.ticks = element_blank(),
legend.justification = c(1, 0),
legend.position = c(0.6, 0.7),
legend.direction = "horizontal") +
guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
title.position = "top", title.hjust = 0.5))
```